REMARKS

This Application has been carefully reviewed in light of the final Office Action dated August 24, 2007 (the "*Office Action*"). At the time of the *Office Action*, Claims 1-69 were pending in the Application with Claims 1-39 being withdrawn. The Examiner rejects Claims 40-51, 60-63, and 65-68 and allows Claims 52-59, 64, and 69. Applicants amend Claims 40-42, 45, 52, and 60-69. Applicants add Claims 70-73. Applicants submit that no new matter is added by these amendments or added claims. Applicants respectfully request reconsideration and favorable action in this case.

Allowable Subject Matter

Applicants note with appreciation the Examiner's indication that Claims 52-59, 64, and 69 are allowed. Claims 52-59, 64, and 69 have not been amended and, therefore, remain in condition for allowance.

Pursuant to 37 C.F.R. § 1.104, Applicants respectfully issue a statement commenting on the Examiner's reasons for allowance. Applicants respectfully disagree with the Examiner's reasons for allowance to the extent that they are inconsistent with applicable case law, statutes, and regulations. Furthermore, Applicants do not admit to any characterization or limitation of the claims, particularly any that are inconsistent with the language of the claims considered in their entirety and including all of their constituent limitations or to any characterization of a reference by the Examiner.

Section 112 Rejections

Claims 40-42, 45, 52 and 60-69 are rejected under 35 U.S.C. § 112, first paragraph, for allegedly failing to comply with the written description requirement. More specifically, the *Office Action* alleges that the limitation "indication from the database" does not support the description of the application's specification. It continues to be Applicants' position that the Examiner is applying the wrong standard. Specifically, the Examiner states that the ""indication from the database" does not support clearly in the description of the application's specification." However, the claims do not need to "support clearly in the description of application's specification. Nevertheless, to advance prosecution of this case, Applicants have amended Claims 40-42, 45, 52, and 60-69 to remove the claim term

"indication from the database." Claims 40-42, 45, 52, and 60-69 now recite "**identifying** one or more dependencies . . ." or an analogous claim limitation. For support in the specification of these claim elements, Applicants refer the Examiner to the following portions of Applicants' specification: Figure 6; Page 11, lines 9-15; Page 12, lines 23-25; Page 13, line 14 through Page 14, line 2; Claims 1-39, as originally filed. Applicants respectfully contend that the limitations stating "identifying one or more dependencies . . ." are fully supported by the specification as originally filed. For at least these reasons, Applicants respectfully contend that Claims 40-42, 45, 52 and 60-69 are patentable under 35 U.S.C. 112.

Section 103 Rejections

The Examiner rejects Claims 40-42, 46-51, 60-62, and 65-67 under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 5,586,328 issued to Caron et al. ("*Caron*") in view of U.S. Patent No. 6,199,063 Colby et al. ("*Colby*"). The Examiner rejects Claims 43-45, 51, 63, and 68 under 35 U.S.C. § 103(a) as being unpatentable over *Caron* in view of *Colby* and further in view of U.S. Patent No. 5,926,819 issued to Doo et al. ("*Doo*"). Applicants request reconsideration and allowance of Claims 40-51, 60-63, and 65-68 for the reasons discussed below.

A. Claims 40, 43-47, 51, 60, 63, 65, and 68

Claims 40, 43-47, 51, 60, 63, 65, and 68 are allowable because the cited references do not disclose the combination of elements recited in Applicants' claims.

For example, the proposed *Caron-Colby* combination fails to disclose, teach, or suggest "recursively querying a database for one or more dependencies of procedural code objects stored in the database," as recited in Applicants' independent Claim 40. In the *Office Action*, the Examiner relies upon *Caron* for disclosure of "dependencies of procedural code objects stored in a database" but acknowledges that *Caron* does not disclose "recursively querying a database." The Examiner states that *Caron* "teaches recursive method for generating dependency code" and that *Colby* "teaches recursively or rewriting the query by a recursive query rewrite call." (*Office Action*, page 4). However, neither of the "recursive" operations disclosed in the references are analogous to Applicants' step of "recursively querying a database . . . ," as recited in Claim 40.

According to *Caron,* "decompilation is initiated by the user performing an edit to a program." (Column 14, lines 53-55). "The compiler next determines whether the edit would actually prevent continuation." (Column 14, lines 66-67). "To make this determination, the compiler at step 208 invokes a method (the "can change method") 230 (FIG. 12) . . . to determine whether the edit to the edited module prevents continuation." (Column 14, line 67 through Column 15, line 4 and Column 15, lines 31-33). "The method 230 is invoked recursively at step 238 for each loaded module to determine whether the effect of "this change" on "this module's" layout or frame prevents continuation." (Column 15, line 65 through Column 16, line 1). Making a determination for each loaded module as to whether a user edit prevents continuation is not analogous to Applicants' step of "recursively querying a database . . . ," as recited in Claim 40. It appears by the Examiner's reliance on *Colby* for disclosure of Applicants' claim language that the Examiner would agree with this position.

However, Applicants contend that *Colby* does not cure the deficiencies of *Caron.* According to *Colby*, a system and method is provided "for answering **a** relational database query." (Column 4, lines 9-10, emphasis added). Specifically, *Colby* discloses:

> **[A] database query** is received. It is then determined whether **that query** can be rewritten in such a manner as to be able to utilize a . . . precomputed view, such that an answer to the rewritten query is equivalent to an answer to the original query. If the query can be rewritten, it is determined whether the rewritten query can more efficiently derive the answer than the original query. **The query** which has the most efficiently derived answer **is the query which is utilized to derive that answer**.

(Column 4, lines 10-20). Thus, only a single query, albeit a rewritten query, is applied to the database. Although *Colby* discloses that the method "attempts to recursively rewrite subqueries" (which is related to the portion of *Colby* cited by the Examiner), with respect to this rewriting of queries *Colby* discloses that "[t]he result is **a rewritten query**, called rewritten query rep in this example." (Column 11, lines 49-60). Recursively rewriting a query to result in a rewritten query that is then used is not analogous to Applicants' step of "recursively querying a database . . .," as recited in Claim 40. Accordingly, Applicants contend that Claim 40 is allowable over the cited references since neither reference discloses operations analogous to Applicants' claim language,

Additionally, Applicants note that the Examiner has not pointed to any reference that discloses "**recursively querying a database for one or more dependencies** of procedural

code objects stored in the database," as recited in Claim 40. Even if *Caron* discloses the "one or more dependencies" and *Colby* discloses the step of "recursively querying a database" (which Applicants do not admit and expressly dispute above), such a piecemeal rejection of Applicants' claim language fails to give credence to each element of Applicants' Claim 40 and to the overall combination of features recited in the claim. The M.P.E.P. provides that "[a]ll words in a claim must be considered in judging the patentability of that claim against the prior art." M.P.E.P. § 2143.03 (citing *In re Wilson*, 424 F.2d 1382, 165 U.S.P.Q. 494, 496 (C.C.P.A. 1970)). Applicant respectfully submits that a rejection of Claim 40 under the *Caron-Colby* combination, in the manner provided by the Examiner, can only result from the piecing together of disjointed portions of unrelated references to reconstruct Applicants' claims. Because the proposed *Caron-Colby* combination does not disclose, teach, or suggest "recursively querying a database for one or more dependencies of procedural code objects stored in the database," Applicants submit that Claim 40 is allowable over the proposed *Caron-Colby* combination.

For at least these reasons, Applicants respectfully request consideration and allowance of Claim 40, together with Claims 43-47 and 51 that depend on Claim 40.

The Examiner also relies on the proposed *Caron-Colby* combination to reject independent Claims 60 and 65. Applicants respectfully submit, however, that the proposed references do not disclose, teach, or suggest the elements recited Applicants' independent Claims 60 and 65. For example, Claim 60 recites "a software module operable to . . . recursively query the database for one or more dependencies of procedural code objects stored in the database." As another example, Claim 65 recites "[a] computer-readable medium encoded with logic operable, when executed on a computer processor, to . . . recursively query the database for one or more dependencies of procedural code objects stored in the database." Thus, for reasons similar to those discussed above with regard to Claim 40, Applicants respectfully submit that Claims 60 and 65 are allowable over the proposed *Caron-Colby* combination. Applicants respectfully request reconsideration and allowance of Claims 60 and 65, together with Claims 63 and 68 that depend on Claims 60 and 65, respectively.

B. Claims 41, 61, and 66

Dependent Claims 41, 61, and 66 depend on Claims 40, 60, and 65, respectively, which Applicants have shown above to be allowable. Accordingly, dependent Claims 41, 61, and 66 are not obvious over the proposed *Caron-Colby* combination at least because Claims 41, 61, and 66 include the limitations of their respective independent claims.

Additionally, Claims 41, 61, and 66 are patentable because they recite additional features and operation not disclosed, taught, or suggested in the proposed *Caron-Colby* combination. For example, Claim 41 incorporates the limitations of independent Claim 40. Thus, Claim 41 requires both "recursively querying a database for one or more dependencies of **procedural code objects** stored in the database" and "recursively querying the database for one or more dependencies of **specifications of object-oriented code objects** stored in the database." Although *Caron* discloses that "three different types of intermodule dependencies are considered: layout, frame, and code" (Column 6, lines 28-30), none of these are analogous to "**one or more dependencies of specifications of object-oriented code objects**," as recited in Applicants' Claim 41. According to *Caron*, "[a] module has a layout dependency if its layout in object code (i.e., the size of storage allocated for each instance of the module at compile time and, in addition, the offset of members within the distance) depends on one or more statements in another module." (Column 6, lines 30-34). "Frame dependencies are commonly created by statements declaring local variables of a procedure" (Column 6, lines 51-52), and "[c]ode dependencies are commonly created by procedure calls, and other statements for which the amount of static or dynamic storage of the module is not altered by a change to another module, but for which a static address may be altered." (Column 7, lines 4-8). However, none of these are disclosed to be "specifications of object-oriented code objects." Accordingly, Applicants submit that the proposed *Caron-Colby* combination, as relied upon by the Examiner, does not disclose, teach, or suggest the combination of elements recited in Applicants' Claim 41.

Additionally, Applicants submit that the *Caron-Colby* combination does not disclose, teach, or suggest "recursively querying the database" to identify the one or more dependencies of the two types disclosed in Applicants' Claim 41. In the *Office Action*, the Examiner's reliance upon *Caron* and *Colby* for disclosure of the additional elements of Claim 41 is almost a verbatim reproduction of the Examiner's rejection of Claim 40. Specifically,

the Examiner states that *Caron* "teaches recursive method for generating dependency code" and that *Colby* "teaches recursively or rewriting the query by a recursive query rewrite call." (*Office Action*, page 5). However, as shown by Applicants above with respect to Claim 40, to the extent that the cited references disclose the features alleged by the Examiner (which Applicants do not admit), the "recursive" operations disclosed in the references are not analogous to Applicants' step of "recursively querying a database . . ." Accordingly, for the same reasons that the proposed references do not disclose "recursively querying the database for one or more dependencies of procedural code objects stored in the database," as recited in Claim 40, Applicants contend that the proposed references do not disclose "recursively querying the database for one or more dependencies of specifications of object-oriented code objects stored in the database," as recited in Claim 41.

Further, even if *Caron* discloses the "one or more dependencies" and *Colby* discloses the step of "recursively querying a database" (which Applicants do not admit and expressly dispute above), such a piecemeal rejection of Applicants' claim language fails to give credence to each element of Applicants' Claim 41 and to the overall combination of features recited in the claim. Applicant respectfully submits that a rejection of Claim 41 under the *Caron-Colby* combination, in the manner provided by the Examiner, can only result from the piecing together of disjointed portions of unrelated references to reconstruct Applicants' claims. Because the proposed *Caron-Colby* combination does not disclose, teach, or suggest "recursively querying a database for one or more dependencies of specifications of object-oriented code objects stored in the database," Applicants submit that Claim 41 is allowable over the proposed *Caron-Colby* combination.

For at least these reasons, Applicants respectfully request consideration and allowance of Claim 41. Claims 61 and 66, which contain certain claim elements that are analogous to those discussed above in Claim 41, are allowable for similar reasons.

C. Claims 42, 62, and 67

Dependent Claims 42, 62, and 67 depend on Claims 40, 60, and 65, respectively, which Applicants have shown above to be allowable. Accordingly, dependent Claims 42, 62, and 67 are not obvious over the proposed *Caron-Colby* combination at least because Claims 42, 62, and 67 include the limitations of their respective independent claims.

Additionally, Claims 42, 62, and 67 are patentable because they recite additional features and operation not disclosed, taught, or suggested in the proposed *Caron-Colby* combination. For example, Claim 42 incorporates the limitations of independent Claim 40. Thus, Claim 42 requires both "recursively querying a database for one or more dependencies of **procedural code objects** stored in the database" and "recursively querying the database for one or more dependencies of **implementations of object-oriented code objects** stored in the database." Although *Caron* discloses that "three different types of intermodule dependencies are considered: layout, frame, and code" (Column 6, lines 28-30), none of these are analogous to "**one or more dependencies of implementations of object-oriented code objects**," as recited in Applicants' Claim 42. According to *Caron*, "[a] module has a layout dependency if its layout in object code (i.e., the size of storage allocated for each instance of the module at compile time and, in addition, the offset of members within the distance) depends on one or more statements in another module." (Column 6, lines 30-34). "Frame dependencies are commonly created by statements declaring local variables of a procedure" (Column 6, lines 51-52), and "[c]ode dependencies are commonly created by procedure calls, and other statements for which the amount of static or dynamic storage of the module is not altered by a change to another module, but for which a static address may be altered." (Column 7, lines 4-8). However, none of these are disclosed to be "implementations of object-oriented code objects." Accordingly, Applicants submit that the proposed *Caron-Colby* combination, as relied upon by the Examiner, does not disclose, teach, or suggest the combination of elements recited in Applicants' Claim 42.

Additionally, Applicants submit that the *Caron-Colby* combination does not disclose, teach, or suggest "recursively querying the database" to identify the one or more dependencies of the two types disclosed in Applicants' Claim 42. In the *Office Action*, the Examiner rejects Claims 41 and 42 together as a group without distinguishing between the different claim elements recited in Claims 41 and 42. Again, the Examiner's reliance upon *Caron* and *Colby* for disclosure of the additional elements of Claim 42 is almost a verbatim reproduction of the Examiner's rejection of Claim 40. Specifically, the Examiner states that *Caron* "teaches recursive method for generating dependency code" and that *Colby* "teaches recursively or rewriting the query by a recursive query rewrite call." (*Office Action*, page 5). However, as shown by Applicants above with respect to Claim 40, to the extent that the cited

references disclose the features alleged by the Examiner (which Applicants do not admit), the "recursive" operations disclosed in the references are not analogous to Applicants' step of "recursively querying a database . . ." Accordingly, for the same reasons that the proposed references do not disclose "recursively querying the database for one or more dependencies of procedural code objects stored in the database," as recited in Claim 40, Applicants contend that the proposed references do not disclose "recursively querying the database for one or more dependencies of **implementations** of object-oriented code objects stored in the database," as recited in Claim 42.

Further, even if *Caron* discloses the "one or more dependencies" and *Colby* discloses the step of "recursively querying a database" (which Applicants do not admit and expressly dispute above), such a piecemeal rejection of Applicants' claim language fails to give credence to each element of Applicants' Claim 42 and to the overall combination of features recited in the claim. Applicant respectfully submits that a rejection of Claim 42 under the *Caron-Colby* combination, in the manner provided by the Examiner, can only result from the piecing together of disjointed portions of unrelated references to reconstruct Applicants' claims. Because the proposed *Caron-Colby* combination does not disclose, teach, or suggest "recursively querying a database for one or more dependencies of **implementations** of object-oriented code objects stored in the database," Applicants submit that Claim 42 is allowable over the proposed *Caron-Colby* combination.

For at least these reasons, Applicants respectfully request consideration and allowance of Claim 42. Claims 62 and 67, which contain certain claim elements that are analogous to those discussed above in Claim 42, are allowable for similar reasons.

D. Claims 48 and 49

Dependent Claims 48 and 49 depend on Claims 40, which Applicants have shown above to be allowable. Accordingly, dependent Claims 48 and 49 are not obvious over the proposed *Caron-Colby* combination at least because Claims 48 and 49 include the limitations of independent Claim 40.

Additionally, Claims 48 and 49 are patentable because they recite additional features and operation not disclosed, taught, or suggested in the proposed *Caron-Colby* combination. For example, the proposed *Caron-Colby* combination does not disclose, teach, or suggest

"identifying one or more cyclic dependencies among code objects stored in the database," as recited in Applicants' Claim 48. As another example, the proposed *Caron-Colby* combination does not disclose, teach, or suggest "utilizing a graph traversal algorithm to identify one or more cyclic dependencies," as recited in Claim 49. In the *Office Action*, the Examiner relies specifically on *Colby* for disclosure of the claim elements of Claims 48 and 49. Specifically, the Examiner cites column 7, lines 4-15 and column 10, lines 18-32 of *Colby*. Applicants contend, however, that neither cited section discloses the elements recited in Applicants' Claims 48 and 49.

The first cited portion of *Colby* discloses a system for answering a database query where the tables "include foreign key-primary key relationships." (Column 7, lines 4-8). According to *Colby*, the "primary key uniquely identifies each row in a database table" and "can be one value from a single column or a combination of values from multiple columns." (Column 7, lines 8-11). By contrast, "[a] foreign key column contains only the values of a primary key column of another table and establishes a many-to-one relationship between the two tables." (Column 7, lines 11-13). "Unlike a primary key column a foreign key column can contain duplicate values." Thus, the cited portion of *Colby* merely discloses that a primary key uniquely identifies each row in a table but that a foreign key establishes a many-to-one relationship between two tables. There is no disclosure in the cited portion that either of the foreign key or primary key are analogous to "one or more cyclic dependencies among code objects stored in the database," as recited in Applicants' Claim 48. Certainly, their is no disclosure in the cited portion of *Colby* of "utilizing a graph traversal algorithm to identify one or more cyclic dependencies," as recited in Claim 49.

The second cited portion of *Colby* discloses a derived table that is created in an example rewritten query. Specifically, *Colby* discloses:

> Note that if day was a primary key of the period table, then no derived table would be needed. A reason for creating the derived table of FIG. 5B is to provide an accurate result. If the sum_dollars_by_day table of FIG. 5A was simply joined with the period table of FIG. 4 under the column day, then the total sum of dollars for each day would be multiplied by the number of times that day appeared in the period of FIG. 4, rather than having the desired effect of having a single sum of the dollar transactions for a given day.

(Column 10, lines 10-26). Thus, *Colby* merely discloses that the primary key results in an accurate summation. It does not at all relate to "cyclic dependencies," as recited in Claims 48

and 49. Thus, this portion of *Colby* also does not disclose, teach, or suggest "identifying one or more cyclic dependencies among code objects stored in the database" or "utilizing a graph traversal algorithm to identify one or more cyclic dependencies," as recited in Claims 48 and 49, respectively.

For at least these reasons, Applicants respectfully request consideration and allowance of Claims 48 and 49.

E. Claim 50

Dependent Claim 50 depends on Claims 40, which Applicants have shown above to be allowable. Accordingly, dependent Claim 50 is not obvious over the proposed *Caron-Colby* combination at least because Claim 50 includes the limitations of independent Claim 40.

Additionally, Claim 50 is patentable because they recite additional features and operation not disclosed, taught, or suggested in the proposed *Caron-Colby* combination. For example, the proposed *Caron-Colby* combination does not disclose, teach, or suggest "generating a dependency graph for code objects stored in the database based at least in part on the dependency information tracking array," as recited in Claim 50. In the *Office Action*, the Examiner states that *Colby* "teaches . . . a dependency graph . . . based at least in part on the dependency information tracking array." Applicants respectfully disagree. The cited portion of *Colby* merely discloses tables that "include foreign key-primary key relationships." (Column 7, lines 4-8). According to *Colby*, the "primary key uniquely identifies each row in a database table" and "can be one value from a single column or a combination of values from multiple columns." (Column 7, lines 8-11). By contrast, "[a] foreign key column contains only the values of a primary key column of another table and establishes a many-to-one relationship between the two tables." (Column 7, lines 11-13). "Unlike a primary key column a foreign key column can contain duplicate values." (Column 7, lines 13-14). According to *Colby*, the primary key results in an accurate summation." (Column 10, lines 10-26). However, the use of primary key and foreign key relationships is not analogous to "generating a dependency graph for code objects stored in the database based at least in part on the dependency information tracking array," as recited in Applicants' Claim 50.

For at least these reasons, Applicants respectfully request consideration and allowance of Claim 50.

New Claims 70-73 are Allowable

New Claims 70-73 have been added and are fully supported by the original specification. No new matter has been added. New Claims 70-73 depend upon independent Claim 40, which is shown above to be allowable. Accordingly, new Claims 70-73 are allowable at least because they incorporate the claim elements recited in independent Claim 40.

Additionally, Claims 70-73 are allowable because they add additional elements that further distinguish the art. For example, Claim 70 recites that "the specifications of object-oriented code objects comprise PL/SQL specifications for a collection of stored functions and procedures identified as a single entity." As another example, Claim 71 recites that "the implementations of object-oriented code objects comprise PL/SQL implementations for a collection of stored functions and procedures identified as a single entity." As shown above with regard to independent Claims 41 and 42, respectively, neither *Caron* nor *Colby* disclose, teach, or suggest "specifications of object-oriented code objects" or "implementations of object-oriented code objects." Certainly then, the proposed *Caron-Colby* combination does not disclose teach or suggest "PL/SQL specifications for a collection of stored functions and procedures identified as a single entity" and "PL/SQL implementations for a collection of stored functions and procedures identified as a single entity," as recited in Claims 70 and 71, respectively.

As still another example, Claim 72 recites "for each of the one or more dependencies identified, determining whether the dependency already occurs in the graph" and "terminating the recursive query of the database upon determining that one of the one or more dependencies already occurs in the graph." These elements are also not disclosed in the cited references. *Caron* merely discloses that "the compiler additionally generates an import table 82 for each module being compiled." (Column 9, lines 44-46). Specifically, where "the module M1 52 includes various statements 62, 70, and 72 which reference members 64, 66, and 68 of other modules M2 54, M3 56, and M4 58 and thereby create dependencies on those modules." (Column 9, lines 46-50). There is no disclosure, however,

of "for each of the one or more dependencies identified, determining whether the dependency already occurs in the graph" and "terminating the recursive query of the database upon determining that one of the one or more dependencies already occurs in the graph," as recited in Claim 72. Neither *Colby* nor *Doo* cure these deficiencies.

As still another example, Claim 73 recites "displaying a dependency graph to a user, the dependency graph generated based at least in part on the dependency information tracking array." With respect to certain similar limitations disclosed in Claim 50, the Examiner states that *Colby* "teaches . . . a dependency graph . . . based at least in part on the dependency information tracking array." Applicants respectfully disagree. The cited portion of *Colby* merely discloses tables that "include foreign key-primary key relationships." (Column 7, lines 4-8). According to *Colby*, the "primary key uniquely identifies each row in a database table" and "can be one value from a single column or a combination of values from multiple columns." (Column 7, lines 8-11). By contrast, "[a] foreign key column contains only the values of a primary key column of another table and establishes a many-to-one relationship between the two tables." (Column 7, lines 11-13). "Unlike a primary key column a foreign key column can contain duplicate values." (Column 7, lines 13-14). According to *Colby*, the primary key results in an accurate summation." (Column 10, lines 10-26). However, the use of primary key and foreign key relationships is not analogous to "a dependency graph . . . based at least in part on the dependency information tracking array," as recited in Applicants' Claim 73. Certainly, it is not analogous to "displaying a dependency graph to a user, the dependency graph generated based at least in part on the dependency information tracking array," as recited in Claim 73.

For at least these reasons, Applicants respectfully submit that new Claims 70-73 are allowable over the prior art.

CONCLUSION

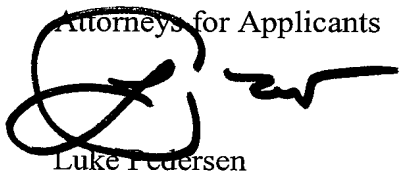
Applicants have made an earnest attempt to place this case in condition for allowance. For the foregoing reasons, and for other apparent reasons, Applicants respectfully request full allowance of all pending Claims. If the Examiner feels that a telephone conference or an interview would advance prosecution of this Application in any manner, the undersigned attorney for Applicants stands ready to conduct such a conference at the convenience of the Examiner.

The Commissioner is authorized to charge the **$810.00 RCE fee,** and to the extent necessary, charge any additional required fees or credit any overpayments to Deposit Account No. 02-0384 of BAKER BOTTS L.L.P.

Respectfully submitted,

BAKER BOTTS L.L.P.

Attorneys for Applicants



Luke Pedersen
Reg. No. 45,003
Tel. 214.953.6655

Date: 10/18/07

CORRESPONDENCE ADDRESS:

Customer Number: **05073**